USING ADAPTIVE THRESHOLDS FOR AUTOMATIC VIDEO CUT DETECTION

Oscar D. Robles[1], Pablo Toharia[1], Ángel Rodríguez[2] and Luis Pastor[1]

[1]Dpto. de Informatica, Estadistica y Telemática.
U. Rey Juan Carlos. C/ Tulipán, s/n. 28933 Móstoles. Madrid. Spain.
{orobles,ptoharia,lpastor}@escet.urjc.es
[2]Dpto. de Tecnología Fotónica.
U. Politécnica de Madrid. Campus de Montegancedo s/n.
28660 Boadilla del Monte. Madrid. Spain
arodri@fi.upm.es

ABSTRACT

This paper presents some automatic shot detection techniques based on color histograms and also on multiresolution histograms. The approaches tested in the submitted runs are:

- H_C4: Differences of color histograms, quantified to 4 classes or bins.
- H_C8: Differences of color histograms, quantified to 8 classes or bins.
- H_C16: Differences of color histograms, quantified to 16 classes or bins.
- H_C32: Differences of color histograms, quantified to 32 classes or bins.
- H_C64: Differences of color histograms, quantified to 64 classes or bins.
- H_CM8: Differences of color histograms, quantified to 8 classes or bins, with redistribution of boundary values.
- H_CM16: Differences of color histograms, quantified to 16 classes or bins, with redistribution of boundary values.
- H_CM32: Differences of color histograms, quantified to 32 classes or bins, with redistribution of boundary values.
- H_CM64: Differences of color histograms, quantified to 64 classes or bins, with redistribution of boundary values.
- HMR_CM64: Differences of multiresolution color histograms, quantified to 64 classes or bins, with redistribution of boundary values.

These runs are mainly focused on cut detection. In this context, the runs with equal redistribution show slightly better results, keeping the recall measure in high values. The main goal of the runs is to study and learn about the behaviour of color histogram differences as a first step in the use of multiresolution color histograms into the adaptive threshold algorithm. To do that, some previous experience on multiresolution histograms has been taken into account.

The techniques herein described improve the behaviour of multiresolution histograms, showing a path to follow to turn them into a powerful tool to work on shot segmentation.

KEY WORDS
CBIR primitives, Video Retrieval, Video Indexing, Shot Segmentation

1 Introduction

One of the main objectives of Content-based Multimedia Retrieval systems is the automatization of the information extraction process from the raw data. When dealing with video data, the first step is to perform a temporal video segmentation in order to make a shot decomposition of the video content. Del Bimbo [1], Brunelli *et al.* [2] and Hanjalic [3] collect extensive reviews of this set of techniques. Depending on the domain of work, these techniques can be classified in non-compressed [4, 5, 6, 7] and compressed video shot segmentation [8, 9, 10].

This work focuses on the exploitation of multiresolution histograms [11] for cut detection using adaptive thresholds. Starting from the Zhang *et al.* description of the technique [12], and taking into account that it does not provide the expected results in all cases, a new implementation has been made, introducing improvements oriented to work with the multiresolution histograms. The main feature of the technique herein described is the high adaptability of the algorithm to a wide range of videos due to the variable threshold managed.

The content of this paper may be broken down into a description of the proposed shot extraction technique (Section 2), then the implementation analysis (Section 3), followed by the results achieved during the tests (Section 4) and the conclusions obtained (Section 5).

2 Shot extraction description

2.1 Multiresolution histograms

The objective is to define a primitive that represents the colour information of an image at different resolution levels. The histogram of the analysis coefficients of the wavelet transform for each colour plane of the multiresolution representation has been chosen. In order to fuse the information proceeding from each resolution level, the values computed at the lower levels are propagated weighting them by a factor of 4^{j-i} , being j the resolution level of the original image and i the considered resolution level (see equation 2):

$$h(k)_{\{R,G,B\}} = \sum_{x,y} I^{(i)}_{\{R,G,B\}}(x,y) \cdot 4^{j-i} \tag{1}$$

$$\forall\,(x,y) \mid I^{(i)}_{\{R,G,B\}}(x,y) = k,\ k = 0,\ldots,255,$$

being $I^{(i)}_{\{R,G,B\}}$ the analysis coefficients of the transformed image at resolution level i and (x,y) the coordinates of each coefficient, the expression of the computed histograms is

$$\hat{h}(k) = \frac{h(k)}{n} \tag{2}$$

where $n = \sum_k h(k)$. It has been considered that information proceeding from lower resolution levels increases the robustness of the discriminant process. This way, we store one histogram per colour plane of the original image, as in the classical approach, but fusing information proceeding from different resolutions of the original image. A deeper study about multirresolution histograms can be found at [11].

2.2 Global strategy

Video cut detection has two main purposes: to delimit the start and the end of the video shots and to process the video content in a more efficient way. The basic idea of video cut detection algorithms is to compute the differences between consecutive frames or groups of frames. Existing techniques differ in the way these differences are computed.

Figure 1 depicts a scheme of the whole process. D_i denotes the difference between the considered frame and

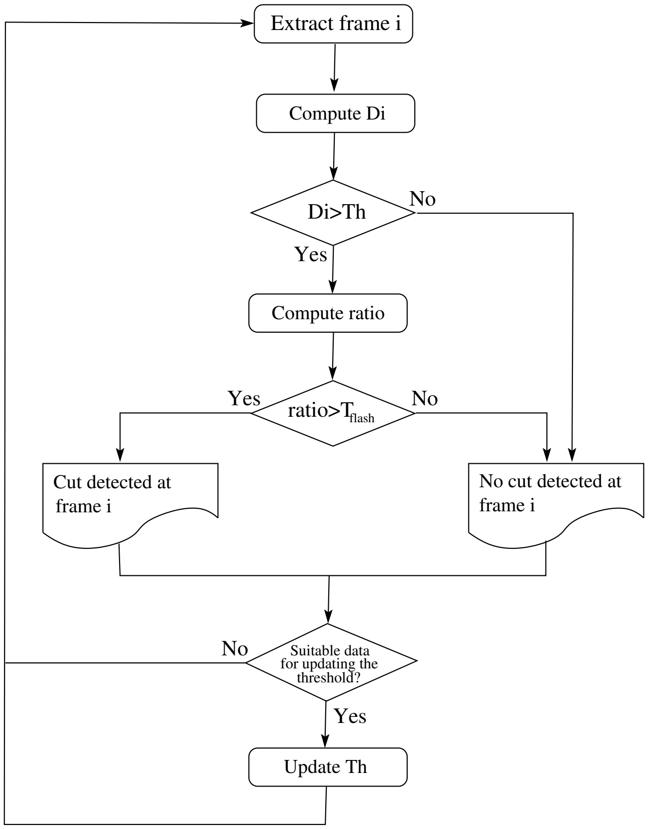


Figure 1. Cut detection algorithm.

the previous one. In our case, the computed D_i difference values are based on several color features in order to make a more exhaustive analysis of the system response. The features implemented have been mean intensity, histograms and multiresolution histograms [11]. A more detailed description of the implemented features can be found further on. A candidate for cut is detected when the values are higher than a dynamically computed threshold Th. The expression of Th is defined by Eq. 3

$$Th \;=\; \text{weight}\frac{\sum_{i=j-W}^{i+W} D(i)}{2W+1} \tag{3}$$

where W is the number of difference values taken into account of the left and right local neighbour windows, i is the frame under consideration and *weight* is a gain factor. Therefore, the threshold is updated for each processed frame.

One of the typical artifacts present in videos is the appearance of flashes that distort the normal analysis of the video signal, because there is no change in the video content but abrupt changes appear in signal intensity. In order to filter out the flashes, a second threshold T_{flash} has been implemented, following the model of Zhang *et al.* [12]. Finally, once the comparisons are performed the threshold Th is recalculated, so that value can be adapted to the new video signal content.

As Figure 1 shows, the cut detection algorithm starts extracting the frame i and computing the difference D_i which is compared against current threshold Th. In case this difference is greater than Th a ratio for detecting flash effects is calculated. If this ratio is greater than threshold T_{flash} a flash is detected. Otherwise, a cut is found. Finally, the current window variance is calculated in order to test whether the data is suitable for updating threshold Th, and in that case a recalculation is needed. These ideas will be explained in depth below.

3 Implementation analysis

Sometimes it happens that the processed difference values D_i vary too much from frame i to next frame $i+1$, as it is the case when, for example, very fast camera movements occur. Therefore, those values that are far away from the sequence recently processed must be discarded. The criteria used to filter these outliers is the variance computed over the sliding window [12]. When the variance V_i is greater than an heuristic predefined threshold T_v, the current threshold Th is not updated.

Flash elimination is done taking into consideration that the appearance of a flash produces an abrupt change in intensity, but unlike real cut edges, the level of the signal comes back to the previous state after one more frame or after a very few ones. The expression that filter flashes is

$$\text{ratio}_{\text{flash}} = \frac{D_{\text{s}}}{D_i}, \tag{4}$$

where D_{s} is the difference between the W frames preceding the current frame and the W ones after it and D_{i} has been defined in Sec. 2.2.

Several color primitives have been tested: mean intensity [12], histograms [12] and multiresolution histograms [11]. The value $\text{ratio}_{\text{flash}}$ has been normalized for each primitive in order to work in the interval $[0,1]$, ideally meaning respectively flash and cut:

$$\begin{cases} \text{ratio}_{\text{flash}} & < & T_{\text{flash}} & \text{Flash detection} \\ \text{ratio}_{\text{flash}} & \geq & T_{\text{flash}} & \text{Cut detection} \end{cases} \tag{5}$$

Each primitive defines different expressions for D_{s}:

- Zhang's implementation includes mean intensity (MI):

$$D_{\text{s}} = \frac{1}{255W} \left| \sum_{k=i-W}^{i-1} MI_k - \sum_{k=i+1}^{i+W} MI_k \right| \tag{6}$$

- Histogram-based primitives:

$$D_{\text{s}} = \frac{1}{2} \sum_{c=0}^{255} |H_{left}(c) - H_{right}(c)| \tag{7}$$

  where

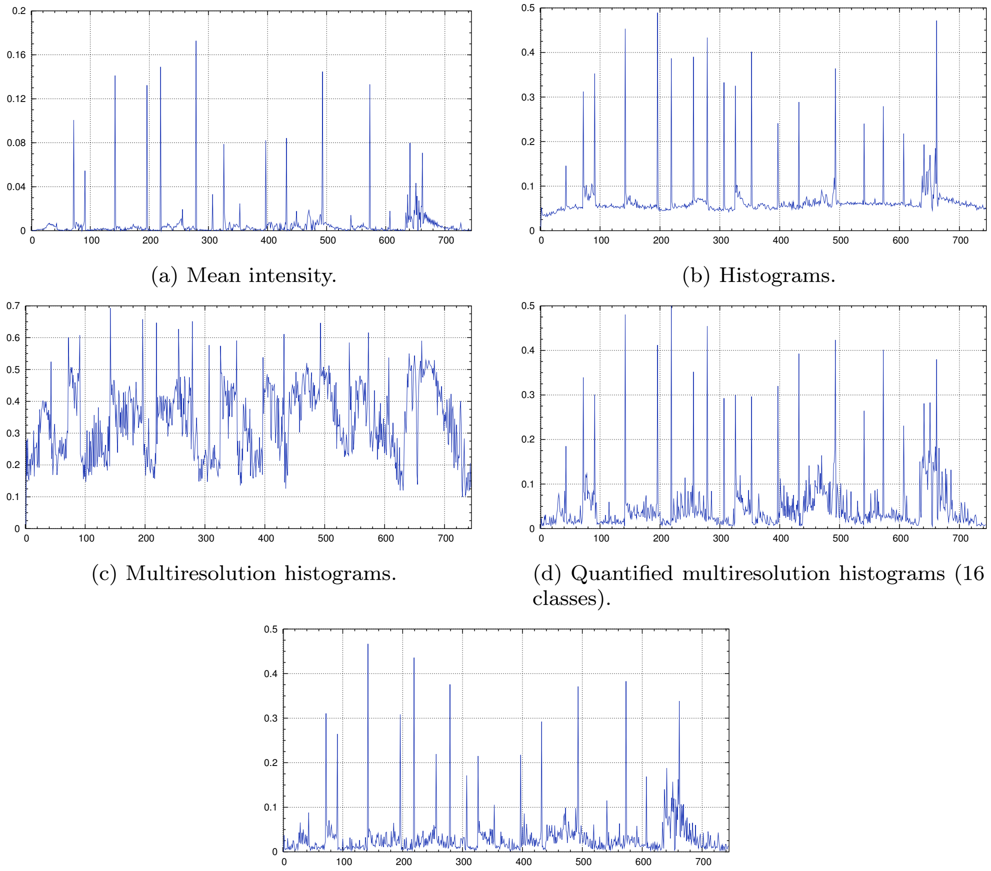
$$\begin{aligned} H_{left}(c) &= \frac{1}{W} \sum_{k=i-W}^{i-1} H_k(c) && (8) \\ H_{right}(c) &= \frac{1}{W} \sum_{k=i+1}^{i+W} H_k(c) && (9) \\ \forall c &= 0, \ldots, 255 \end{aligned}$$

The interval $[0,1]$ has been equally subdivided to assign $D_{\text{s}} < 0.5$ to flashes.

Histograms usually present a problem when comparing distributions concentrated around near but not exactly equal values. The result of the comparison produces high difference values although the appearance of the images that generate the histograms is very similar. To reduce this effect a histogram quantification has been performed, grouping color values to generate new classes:

$$H_q(i) = \sum_{j=iT_c}^{iT_c+T_c-1} H(j), \ \forall i \in [0, N_c - 1] \tag{10}$$

where H_q is the quantified histogram, H is the original histogram, T_c is the size of the new classes and N_c is the number of classes ($N_c \cdot T_c = 256$ for 8 bit cases).



(a) Mean intensity.

(b) Histograms.

(c) Multiresolution histograms.

(d) Quantified multiresolution histograms (16 classes).

(e) Quantified, equally redistributed multiresolution histograms (16 classes).

Figure 2. Comparison of graphs showing number of frames against D_i values.

While working with the quantification, the same problem can appear around the boundaries of the classes. It can be solved equally redistributing edge levels between neighbour classes. It can be achieved computing the value c_i:

$$c_i = \frac{\sum_{j=iTc-\varepsilon}^{iT_c+\varepsilon} H(j)}{2\varepsilon} \tag{11}$$

so the frequency of class i on the resultant quantified histogram with equal redistribution $H_q^e(i)$ will be:

$$H_q^e(i) = c_i + c_{i+1} + \sum_{j=iT_c+\varepsilon}^{iT_c+T_c-1-\varepsilon} H(j) \tag{12}$$

where the summatory comes from equation 10. It must be noticed that the first and last classes of the histogram are not considered on equation 12, since c_i is not considered when $i = 0$ and c_{i+1} is not considered when $i = N_c - 1$.

Figure 2 shows a comparison among the distributions of D_i values computed over one of the videos used in the experiments by the features previously described. Each graph presents the number of frames against D_i values. It can be seen that around the frames 260 and 300, the Figure 2(a) does not show some peaks that apperar in Figure 2(b) and are expected to be cuts. On other side, Figure 2(c) shows unreal differences, while Figure 2(d) shows a smoother graph.

Once the cuts have been detected, a formalization of the edges extraction is written to an XML description of the extracted shots [14]. In order to summarize the content of a shot, we have chosen a key frame per shot, specifying the beginning, the end, and the key frame for every detected shot.

4 Experimental Results

4.1 Experiments Setup

The main objectives of the tests are to measure and analyze the recall and precision values of the implemented features with and without the dynamic threshold. We use the classical definition of recall and precision:

$$\begin{align} \text{Recall:} &= \frac{\text{True positives}}{\text{True positives} + \text{False negatives}} \tag{13} \\ \text{Precision:} &= \frac{\text{True positives}}{\text{True positives} + \text{False positives}} \tag{14} \\ & \tag{15} \end{align}$$

The results shown in the following sections correspond to more than the 10 runs evaluated by the TRECVID 2004 team. This evaluation considers short gradual transitions as cuts, whereas our evaluations do not include this point. That is the reason why our own evaluation of them is included, as well as the ground truth data provided by the TRECVID team.

All the tools involved in the developed software are free distribution tools, like vs. 2.4.18-14 Linux operating system, vs. 3.2.7 of the GCC GNU compiler [15], vs. 0.4.0 of the MPEG-2 video stream decoder LIBMPEG-2 [16] and vs. 2.4.23 of the LIBXML2 library for processing XML files [17].

4.2 Results analysis

Table 1 shows the recall and precision values, as returned by the TRECVID 2004 team, for the cut detection task. The suffix notation used in the method column in the tables is the following: MI —mean intensity—, H —histogram—, C —quantified—, MR —multiresolution—, E —equally redistributed— and number —number of classes—.

Método	**Recall**	**Precision**
H_C4	0.883	0.748
H_C8	0.875	0.811
H_C16	0.868	0.855
H_C32	0.845	0.884
H_C64	0.792	0.895
H_CM8	0.869	0.759
H_CM16	0.864	0.815
H_CM32	0.860	0.839
H_CM64	0.851	0.849
HMR_CM64	0.759	0.858

Table 1. Precision and recall evaluated by TRECVID 2004.

Table 1 shows how the quantification (labels H_C4 to H_C64) improves the precision values while produces lower values of recall. It can also be seen that the use of the equal redistribution (labels H_CM8 to H_CM64 in the table) keeps both the recall and precision in high values. Finally, a last run was submitted to test both techniques applied to the multiresolution histograms (label HMR_CM64 in the table).

To acomplish a proper evaluation of the experiments not submitted to the TRECVID and finish all the study, it must be noticed that the TRECVID evaluation assumes short graduals as cuts. On one hand, our own evaluation methods do not consider this matter, on the other hand the techniques herein analyzed have not been developed to give a suitable answer to that point. For that reason, Table 2 shows the results achieved by all the methods, repeating in our own evaluation those also shown in Table 1.

Método	**Recall**	**Precision**
MI	0.873	0.248
H	0.516	0.781
H_C4	0.932	0.672
H_C8	0.931	0.734
H_C16	0.922	0.773
H_C32	0.901	0.801
H_C64	0.843	0.810
H_CM4	0.924	0.558
H_CM8	0.925	0.687
H_CM16	0.920	0.738
H_CM32	0.919	0.762
H_CM64	0.912	0.775
HMR	0.002	0.073
HMR_C4	0.915	0.374
HMR_C8	0.903	0.490
HMR_C16	0.871	0.688
HMR_C32	0.766	0.775
HMR_C64	0.431	0.760
HMR_CM4	0.902	0.289
HMR_CM8	0.907	0.442
HMR_CM16	0.883	0.652
HMR_CM32	0.874	0.726
HMR_CM64	0.811	0.778

Table 2. Own evaluation of TRECVID 2004 data.

Excluding the short graduals of the evaluation causes an increase of the recall, since these techniques do not recognize them as a cut, and a decrease of the precision, due to the fact that the TRECVID evaluation classifies them as a hit instead as a false positive. Appart from that, the same tendency shown in Table 1 can be observed in Table 2. The implementation of the original techniques of Zhang's algorithm have also been tested, with poor precision results in the case of mean intensity (label MI) and a low value of recall in the case of histograms (label H). It can also be seen that the quantification and the redistribution of the values really improve the results achieved by the multiresolution histograms (label HMR).

Table 3 shows data about the processing complexity of the runs submitted to the TRECVID. The decode time includes a rescaling stage for every frame. It has not been included into the segmentation time because it has been implemented using a general algorithm based on bicubic interpolation whithout considering any optimization in terms of performance.

Figure 3 shows the combination of recall and precision values for our own evaluation of the runs shown on Table 2. The figures show the high stability of the methods, since it can be seen that all the values are grouped around the mean values.

Tables 1 and 2 and Figure 3 confirm that the lower the number of classes is, the higher the recall is, but with a loss of precision. The techniques implemented keep both precision and recall in high values.

Method	Total decode time	Total segmentation time	Processor type and speed
H_C4	5492.36	430.26	Pentium IV 3GHz
H_C8	5496.47	430.2	Pentium IV 3GHz
H_C16	5470.96	431.58	Pentium IV 3GHz
H_C32	5484.86	431.02	Pentium IV 3GHz
H_C64	5584.95	435.2	Pentium IV 3GHz
H_CM8	5477.48	434.81	Pentium IV 3GHz
H_CM16	5481.69	435.37	Pentium IV 3GHz
H_CM32	5583.42	435.38	Pentium IV 3GHz
H_CM64	5500.85	443.49	Pentium IV 3GHz
HMR_CM64	5511.29	2912.84	Pentium IV 3GHz

Table 3. Processing complexity.

5 Conclusions and ongoing work

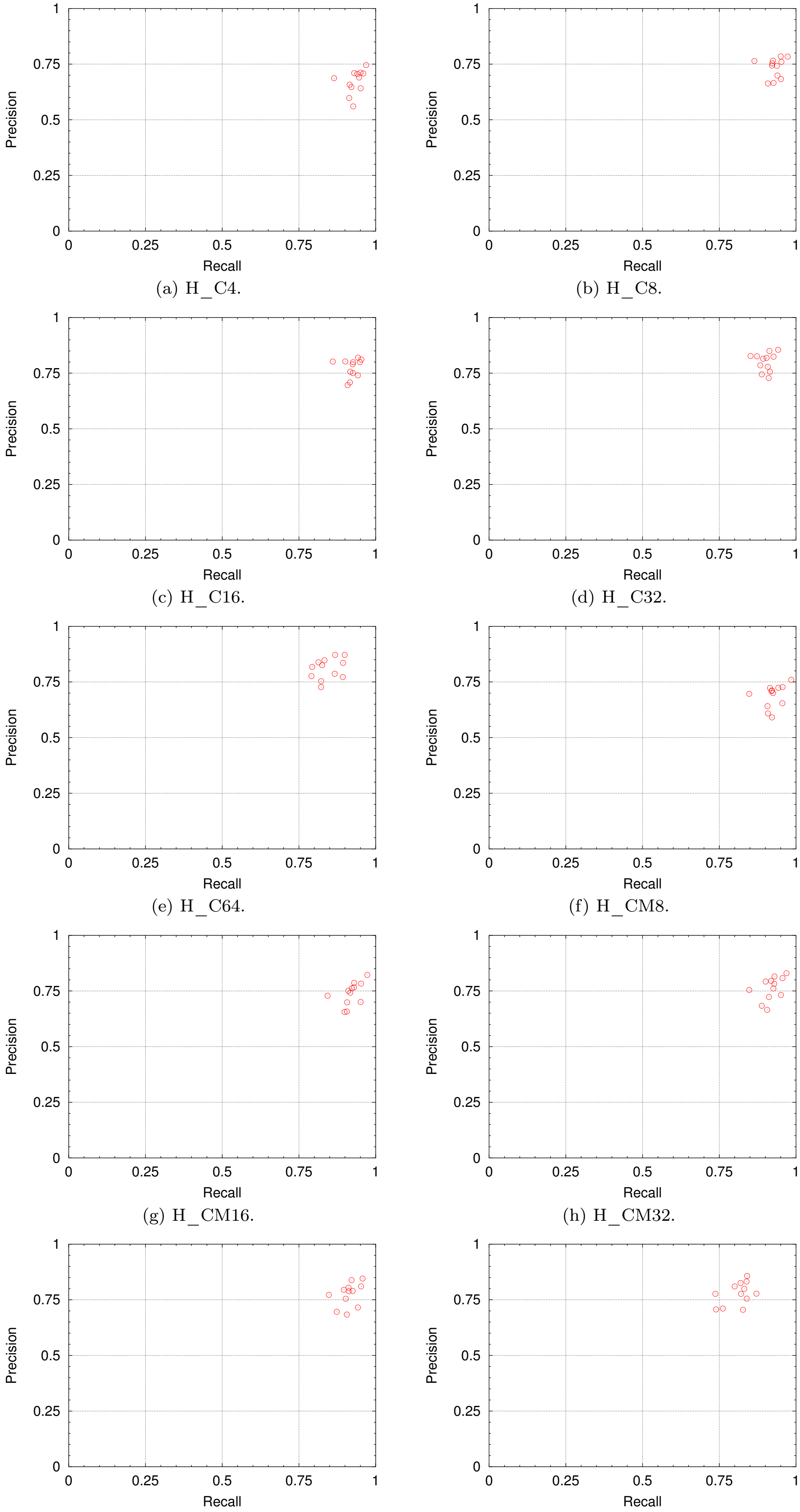
In this paper, some autommatic shot detection techniques based on color histograms and multiresolution histograms are presented. The techniques applied to achieve interesting results with the color multiresolution histograms have been also used on color histograms to test their behaviour. The results have shown that this techniques improve the behaviour of the multiresolution histograms, although it has to be done more work to obtain better results.

Different approaches, like the application of different filters, will probably provide smoother difference graphs than those shown on Figure 2 and making more clear the disrimination of peaks corresponding to cuts. The use of local multiresolution histograms [11] will also help to build a more powerful tool to detect cuts, and perhaps some graduals. Appart from that, a combination of the primitives may improve the precision and recall measures.

More work will be focused on a more efficient implementation of the threshold Th updating. Alternatives to variance computation will be studied.

Acknowledgments

This work has been partially funded by the Spanish Commission for Science and Technology (grants CICYT TIC2003-08933-C02-01).



(a) H_C4.

(b) H_C8.

(c) H_C16.

(d) H_C32.

(e) H_C64.

(f) H_CM8.

(g) H_CM16.

(h) H_CM32.

(i) H_CM64.

(j) HMR_CM64.

Figure 3. Recall-precision graphs of our own evaluation.

References

[1] Alberto del Bimbo. *Visual Information Retrieval*. Morgan Kaufmann Publishers, San Francisco, California, 1999. ISBN 1-55860-624-6.

[2] R. Brunelli, O. Mich, and C. M. Modena. A survey on video indexing. *Journal of Visual Communication and Image Representation*, 10:78–112, 1999.

[3] Alan Hanjalic. Shot-boundary detection: Unraveled and resolved? *IEEE Transactions on Circuits and Systems for Video Technology*, 12(2):90–105, February 2002.

[4] Sara Porter, Majid Mirmehdi, and Barry Thomas. Temporal video segmentation and classification of edit effects. *Image and Vision Computing*, 21(13–14):1097–1106, December 2003.

[5] Min Gyo Chung, Hyeokman Kim, and S. Moon-Ho Song. A scene boundary detection method. In *Proceedings of the International Conference on Image Processing 2000, ICIP 00*, volume 3, pages 933–936, Vancouver, September 2000. IEEE Computer Society.

[6] G. Valencia, J. A. Rodríguez, C. Urdiales, and F. Sandoval. Color-based video segmentation using interlinked irregular pyramids. *Pattern Recognition*, 37(2):377–380, February 2004.

[7] Rozenn Dahyot, Niall Rea, and Anil Kokaram. Sport video shot segmentation and classification. In Tonradj Ebrahimi and Thomas Sikora, editors, *Visual Communications and Image Processing 2003*, volume 5150, pages 404–413, Univ. of Italian Switzerland (USI), Lugano, Switzerland, July 2003. SPIE. ISBN 0-8194-5023-5.

[8] Sameer Antani, Rangachar Kasturi, and Ramesh Jain. A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. *Pattern Recognition*, 35(4):945–965, April 2002.

[9] Robert A. Joyce and Bede Liu. Temporal segmentation of video using frame and histogram-space. In *Proceedings of the International Conference on Image Processing 2000, ICIP 00*, volume 3, pages 941–944, Vancouver, September 2000. IEEE Computer Society.

[10] Hongjiang Zhang. Video content analysis and retrieval. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, chapter 5.5, pages 945–977. World Scientific Publishing Company, 1998.

[11] Oscar D. Robles, Angel Rodríguez, and M. Luisa Córdoba. A study about multiresolution primitives for content-based image retrieval using wavelets. In M. H. Hamza, editor, *IASTED International Conference On Visualization, Imaging, and Image Processing (VIIP 2001)*, pages 506–511, Marbella, Spain, September 2001. IASTED, ACTA Press. ISBN 0-88986-309-1.

[12] Dong Zhang, Wei Qi, and Hong Jiang Zhang. A new shot boundary detection algorithm. In Heung-Yeung Shum, Mark Liao, and Shih-Fu Chang, editors, *IEEE Pacific Rim Conference on Multimedia*, volume 2195, pages 63–70. IEEE, Springer, October 2001.

[13] Angel Rodríguez, Oscar D. Robles, and Luis Pastor. New features for Content-Based Image Retrieval using wavelets. In Fernando Muge, Rogério Caldas Pinto, and Moisés Piedade, editors, *V Ibero-american Simposium on Pattern Recognition, SIARP 2000*, pages 517–528, Lisbon, Portugal, September 2000. ISBN 972-97711-1-1.

[14] Extensible Markup Language (XML) 1.0 (Second Edition). Web, oct 2000. W3C Recommendation. `http://www.w3.org/TR/REC-xml`

[15] GNU. `http://www.gnu.org`

[16] Free MPEG-2 video stream decoder. Web. `http://libmpeg2.sourceforge.net`

[17] Gnome Project. Gnome XML C parser and toolkit. Web. `http://www.xmlsoft.org`